REMARKS

In the above identified Office Action, the Examiner objected to Claims 6, 8 and 10 for the use of the word "analysing" instead of "analyzing". Claims 1 – 5, 8 - 20, 23, 24 and 27 - 34 were rejected under 35 U.S.C. §102(b) as being anticipated by Gish. Claims 6, 7 21, 22, 25 and 26 were rejected under 35 U.S.C. §103(a) as being unpatentable over Gish in view of Nakajima.

In reviewing the Specification, Applicants noticed a few typographical errors and corrected them. Applicants further amended Claims 5, 8 and 10 to replace the word "analysing" with "analyzing." Applicants believe that the objection has been overcome and kindly request its withdrawal.

By this amendment, Claims 1 – 34 remain pending in the Application. For the reasons stated more fully below, Applicants submit that the pending claims are allowable over the applied references. Hence, reconsideration, allowance and passage to issue are respectfully requested.

The invention is set forth in claims of varying scopes of which Claim 1 is illustrative.

> 1. A method of executing code in a client server environment comprising:
> identifying an input object on a client system, the input object identifying code for executing on a server;
> ***processing the input object to identify the code for executing on a server***;
> ***generating, in response to identifying the code for executing on a server, code for accessing the code for executing on a server***;
> ***processing the generated code to determine a server for executing the code for executing on a server, each code for executing on a server being able to execute on a particular server***;
> ***enabling the determined server to access the code for executing on a server***;
> ***identifying, based on the accessed code for executing on a server, a client application for allowing the determined server to interact with the***

> ***client system during processing of the code for executing on a server***; and
> processing the code for executing on a server on the determined server. (Emphasis added.)

The Examiner rejected Claims 1 – 5, 8 – 20, 23, 24 and 27 - 34 under 35 U.S.C. §102(b) as being anticipated by Gish. Applicants respectfully disagree.

Gish purports to teach an enterprise computing manager in which an application is composed of a client (front end) program which communicates utilizing a network with a server (back end) program. The client and server programs are loosely coupled and exchange information using the network. The client program is composed of a User Interface (UI) and an object-oriented framework (Presentation Engine (PE) framework). The UI exchanges data messages with the framework. The framework is designed to handle two types of messages: (1) messages from the UI, and (2) messages from the server (back end) program via the network. The framework includes a component, the mediator which manages messages coming into and going out of the framework.

The Examiner asserted that Gish teaches an input object identifying code for executing on a server in col. 19, lines 13 – 19. Applicants disagree.

In col. 19, lines 13 – 22 Gish discloses:

> When application execution is initiated, the client node begins to interpret the PE 700 (FIG. 7) it has received from the server node 710. The PE 700 is a framework (which includes an User Interface (UI) which can include a graphical UI (GUI)) and an instance of a communication library 720 implemented in Java. Once it starts up, the PE 700 opens a socket connection to the server node 710 utilizing the server port number it was supplied when the server app manager 650 started the back end process 730.

Thus, the framework (i.e., PE 700) does not process an input object that identifies code to be executed on a server. Rather, the framework sends commands to the server for the server to open a socket.

In short, Gish does not teach, show or suggest the steps of ***processing the input object to identify the code for executing on a server***; ***generating, in response to identifying the code for executing on a server, code for accessing the code for executing on a server***; ***processing the generated code to determine a server for executing the code for executing on a server, each code for executing on a server being able to execute on a particular server***; ***enabling the determined server to access the code for executing on a server***; and ***identifying, based on the accessed code for executing on a server, a client application for allowing the determined server to interact with the client system during processing of the code for executing on a server*** as in Claim 1.

Regarding independent Claim 12, the Examiner asserted that Gish clearly discloses an extensible mechanism for executing code in a client server environment in col. 18, lines 9 and 10. Applicants respectfully disagree.

In col. 18, lines 9 and 10, Gish discloses:

> The client and server nodes communicate utilizing the web technologies in an Internet, Intranet or other network environment.

Thus, it can be said that Gish discloses a mechanism for executing code in a client server environment. However, Gish does not teach an ***extensible*** mechanism for executing code in a client server environment as in the claimed invention.

Note that ***extensible***, as defined in the Specification and as used in the claims, means being able to add new functionalities and/or modify existing functionality while minimizing impact to existing system functions (see paragraphs [0052] and [0055] of the Specification).

The Examiner further asserted that Gish teaches a view mechanism for processing an input object identifying code for executing on a server in col. 19, lines 13 – 19. Again, Applicants disagree.

In the paragraph in col. 19, lines 13 – 22 (see the reproduced paragraph above) even if the framework (i.e., PE 700) is the view mechanism, it does not process an input object that identifies code to be executed on a server. Rather, it sends commands to the server for the server to open a socket.

Further, the Examiner asserted that Gish teaches outputting a deployable object in col. 28, lines 46 – 63 and then stated that Gish teaches a server mechanism for processing the deployable object to determine a particular server for executing the code in col. 27, lines 61-62.

In col. 28, lines 46 – 63, Gish discloses:

> When a user interface event occurs--for example, when the user enters a first name in the text field and clicks a Search button--the user interface passes a message to the Presentation Engine. The Presentation Engine either sends the message to its data model for processing by the client, or passes the message to the server for processing by the server program. The Presentation Engine determines where on the client a message is handled based on how you have registered message handlers. When the server program sends a message with data to the client, the Presentation Engine displays the result.
>
> The exchange of messages between client and server is handled through the ICE-T Communication Libraries in the ICE-T Communication Layer. When the client program terminates, the Application Manager closes the socket connection to the client and terminates any server processes it started.

In the reproduced passages reproduced immediately above, only messages are being passed from clients to servers and vice versa. Thus, the deployable object that the Examiner asserted to be taught by Gish can only be a message.

In col. 27, lines 61 and 62, Gish discloses:

> The applet contains the URL of the server holding the application components and the application name.

In this passage, however, Gish discloses an applet and a URL. Therefore, the deployable object cannot be a message, which the Examiner has asserted in col. 28, lines 46 – 63 as being a deployable object.

Therefore, Applicants submit that Gish does not teach ***a view mechanism for processing an input object identifying code for executing on a server and outputting a deployable object***; ***a server mechanism for processing the deployable object to determine a particular server for executing the code and to enable the deployable object to execute on the particular server, said second mechanism outputting a launchable object*** as in Claim 12.

Regarding independent Claim 29, the Examiner stated that Gish teaches "said computer system executing an integrated development environment (IDE) for generating code for executing in a client server environment." Applicants disagree.

As mentioned above, Gish purports to teach an enterprise computing manager in which an application is composed of a client (front end) program which communicates utilizing a network with a server (back end) program. However, Gish does not teach a system executing an **IDE**. Nor does Gish teach an **extensible** mechanism.

Since Gish does not teach an IDE, then Gish cannot teach adapting the IDE to ***process an input object identifying code for executing on a server, said processing using a view list of at least one input object element, each input object element processing a type of code identified by the input object to output a deployable object***; ***process the deployable object using a server list of at least one server element to determine a server for executing the code, each server element enabling the deployable object to execute on a particular server and outputting a launchable object***; and ***process the launchable object using a launcher list of at least one client***

element to determine a client for launching the code on the particular server.

Regarding independent Claim 32, Gish does not teach a view list of at least one input object element or a server list of at least one server element to determine a server for executing the code or a launcher list of at least one client element to determine a client for launching the code on the particular server.

The Examiner rejected Claims 6, 7, 21, 22, 25 and 26 under 35 U.S.C. §103(a) as being unpatentable over Gish in view of Nakajima. Applicants disagree.

Claims 6 and 21 depend directly on independent Claims 1 and 12, respectively and contain similar limitations.

As mentioned above, Claims 1 and 12 are not anticipated by Gish. Further, Nakajima does not teach what the Examiner asserted that it teaches.

Nakajima purports to teach a method for client-side handling of extensions originally written for servers. In accordance with the teachings of Nakajima, when a local browser is notified that an extension is local, the browser creates a moniker for interfacing with the extension. Although local, the moniker appears to the extension to be a server process, whereby the extension executes its function on user data and provides a result to the moniker. The moniker returns the result to the browser for display thereof. The method then enables extensions developed for remote servers to be locally executed in a client machine, while using the local browser as a user interface.

The Examiner asserted that Nakajima discloses the step of processing the input object comprises: analyzing the input object to determine an input object element for processing the input object in col. 3, lines 55 – 60; and processing the input object using determined input object element in col. 3, lines 55 - 66. The Examiner then interpreted the passage showing "how an input object element (code) is processed (formatted), and how the user via the use of an input device processes the input object elements."

In col. 3, lines 46 – 66, Nakajima discloses:

> By way of background, FIG. 2 shows the state of the prior art. As is well known, the client computer system 20 can include at least one conventional application program 40 which, for example, can receive instructions from the user-input device 30 (FIG. 1) to read information from the database 38. Also loaded in the memory 24 (but not shown herein for purposes of simplicity) is an operating system. In conjunction with the operating system, the application program 40 reads the database 38 by placing API calls to appropriate WIN32 APIs 42. The application program 40 typically includes code 44 for processing (e.g., formatting) the data read from the database 38, and also includes a user-interface 46 for outputting information related to the processed data for viewing by the user via the output device 34.
>
> Moreover, in FIG. 2, the client computer system 20 has loaded in its memory 24 a browser 50 which provides the user with a graphical interface to the Internet 52. Communication over the Internet is via the Hypertext Transfer Protocol (HTTP). To simplify handling of the HTTP requirements, the browser 50 incorporates OLE object monikers, shown in FIG. 2 as a URL moniker 54.

But note that nothing in the reproduced passages teaches **analyzing an input object** to determine an input object element for processing the input object.

Thus, even if the Examiner's assertions regarding the teachings of Gish and Nakajima were correct, the combination of Gish and Nakajima would not teach the claimed invention.

Nonetheless, to combine the teachings of two or more references, there has to be a teaching, suggestion or motivation to do so.

In this case, Gish purports to teach a mechanism for executing client-side code on a client machine and server-side code on a server machine wherein

messages are sent between the client and server for coordination. By contrast, Nakajima purports to teach executing server-side code on a client machine. Therefore, there is no reason for anyone to combine the teachings of the two references absent some suggestion or teaching to do so.

In any event, Applicants submit that independent Claims 1, 12, 29 and 32 as well as their dependent claims, are allowable over the cited references. Thus, Applicants once more respectfully request reconsideration, allowance and passage to issue of the claims in the application.

Respectfully Submitted

By: Volel Emile
Attorney for Applicants
Registration No. 39,969
(512) 306-7969